Semi-formal assertion generation

to address consistency/error-handling in the PRiME framework

Red: assertions to ensure control flows

Green: assertions to ensure boundary correctness conditions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| api identifier | | api description | api parameters | api call correctness assertions |
| app\_reg | | Register an application with RTM | *a*: application | * *a is not registered* |
| app\_dereg | | De-register an application with RTM | *a*: application | * *a is registered* * *no knob registered with a* * *no monitor registered with a* |
| application knob | | | | |
| app\_knob\_reg | Register an application-level knob | | *a*: application | * *a is registered* * *ak is not registered for a* * *min < max* |
| *ak*: app knob |
| *min*: min value |
| *max*: max value |
| RTM\_app\_knob\_set | Set the setting of an application-level knob | | *a*: application | * *a is registered* * *ak is registered with a: (a, ak)* * *min\*<= value <= max\**   *\*min and max of (a, ak)* |
| *ak*: app knob |
| *value*: set value |
| app\_knob\_get | Fetch the setting of an application-level knob | | *a*: application | * *ak is registered with a* * *(a, ak) has a value* |
| *ak*: app knob |
| app\_knob\_set\_min | Update the minimum allowed value of an application-level knob | | *a*: application | * *min is set for (a, ak)* * *min <= max\**   *\* max of (a, ak)*   * *min <= value\**   *\* if there is a value set for the (a, ak)* |
| *ak*: app knob |
| *min*: set value |
| app\_knob\_set\_max | Update the maximum allowed value of an application-level knob | | *a*: application | * *max is set for (a, ak)* * *max >= min\**   *\* min of (a, ak)*   * *max >= value\**   *\* if there is a value set for the (a, ak)* |
| *ak*: app knob |
| *max*: set value |
| app\_knob\_dereg | De-register an application-level knob with RTM | | *a*: application | * *ak is registered with a* |
| *ak*: app knob |
| application monitor | | | | |
| app\_mon\_reg | Register an application-level monitor | | *a*: application | * *a is registered* * *am is not registered for a* * *min < max* * *weight ≠ 0*   *or min ≠ MIN\_INFINITY*  *or max ≠ MAX\_INFINITY* |
| *am*: app monitor |
| *min*: min value |
| *max*: max value |
| *weight*: weight value |
| app\_mon\_set | Update application-level monitor’s value | | *a*: application | * *a is registered* * *am is registered with a: (a, am)* * *min\*<= value <= max\**   *\*min and max of (a, am)* |
| *am*: app monitor |
| *value*: set value |
| app\_mon\_set\_min | Update the minimum allowed value of an application-level monitor | | *a*: application | * *am is registered with a* * *min <= max\**   *\* max of (a, am)*   * *min <= value\**   *\* if there is a value set for the (a, am)*   * *weight ≠ 0*   *or min ≠ MIN\_INFINITY*  *or max ≠ MAX\_INFINITY* |
| *am*: app monitor |
| *min*: set value |
| app\_mon\_set\_max | Update the maximum allowed value of an application-level monitor | | *a*: application | * *am is registered with a* * *max >= min\**   *\* min of (a, am)*   * *max >= value\**   *\* if there is a value set for the (a, am)*   * *weight ≠ 0*   *or min ≠ MIN\_INFINITY*  *or max ≠ MAX\_INFINITY* |
| *am*: app monitor |
| *max*: set value |
| app\_mon\_set\_weight | Update the importance of an application-level monitor | | *a*: application | * *am is registered with a* * *weight ≠ 0*   *or min ≠ MIN\_INFINITY*  *or max ≠ MAX\_INFINITY* |
| *am*: app monitor |
| *weight*: set value |
| app\_mon\_dereg | De-register an application-level monitor | | *a*: application | * *am is registered with a* |
| *am*: app knob |
| device knob | | | | |
| dev\_knob\_reg | Register all device-level knobs of a device. | | *dks*: set of dev knobs |  |
| dev\_knob\_set\_type | Return type (voltage, freq, etc.) of a device-level knob. | | dk: dev knob | * *dk is registered* * *t is a valid type* |
| t: set type |
| dev\_knob\_set | Return type (voltage, freq, etc.) of a device-level knob. | | dk: dev knob | * *dk is registered* |
| value: set value |
| dev\_knob\_dereg | De-register all device-level knobs of a device. | | dk: dev knob | * *dk is registered* |
| Device monitor | | | | |
| dev\_mon\_reg | Register all device-level monitors that exist | | *dms*: set of dev monitors |  |
| dev\_mon\_set\_type | Return type (power, slack, etc.) of a device-level monitor | | dm: dev monitor | * *dm is registered* * *t is a valid type* |
| t: set type |
| dev\_mon\_get | Fetch the setting of a device-level monitor | | dm: dev monitor | * *dm is registered* |
| value: set value |
| dev\_mon\_dereg | De-register all device-level monitors that exist | | dm: dev monitor | * *dm is registered* |